Relatório competição 2

Lucas Vitor de Souza
Fevereiro 2023

1 Análise exploratória dos dados

Os dados de treino foram utilizados para a verificação das métricas e para medir o desempenho da classificação, contudo tanto a base de dados de treino como teste passaram pela mesma análise de dados antes de se iniciar o treinamento.

A princípio foram importados ambos os datasets, Treino e Teste, em formato .csv utilizando-se da biblioteca pandas. A primeira operação foi imprimir o dataset de treino para verificação das features, são 3 colunas com 7500 linhas, tais quais variam seus tipos de dados em int64 e object, como mostrado a seguir:

```
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          7500 non-null   int64
 1   misogynous  7500 non-null   int64
 2   text        7500 non-null   object
dtypes: int64(2), object(1)
```

Os dados contidos nesse dataset podem ser divididos em Numéricos e Categóricos, com variáveis numericas discretas, pode ser observado também através dessa tabela, que não existem features que possuem valores Null (NaN). Dessa forma não foi necessário o tratamento dos dados do tipo NaN.

O Passo seguinte foi verificar o tipo de dado object do nosso data set, para isso, o mesmo foi impresso para verificar como era a formatação desses dados, e foi constatado que os dados se tratavam de strings que não apresentavam normalização, ou seja, as strings misturavam letras e numeros, maiúsculas e minúsculas, pontuações e variás acentuações gráficas, como mostrado a seguir.

```
id  misogynous                                                 text
0    0           0  ME: WORKING REMOTELY DOING THE CHORES DOING GR...
1    1           1  imgilip.com Divorce Childrens well-being Woman...
2    2           1        A GIRL WHO SHOWS A LOT, HAS LITTLE TO OFFER
3    3           1  Feminist: we can do everything that men do Men...
4    4           1  r/ConservativeMemes Posted by u/undue-influenc...
5    5           1   This is not Feminism. ìš°> >3^ This is Feminism.
6    6           0     you have the personality of a carpeted kitchen
7    7           1  The entire halftime show was a form of female ...
8    8           1  hot blonde going thru a street å ² No way i'n ...
```

Dessa forma, foi necessário um tratamento dessas strings para que o resultado inicial obtido na competição do Kaggle (0.69368) sem normalização, fosse melhorado até chegar no terceiro lugar do ranking da competição, com uma pontuação de (0.78780). O tratamento feito foi a remoção da pontuação, das stopwords e a normalização das strings, deixando todas em lowercase.

2 Pré-processamentos realizados

Com a análise exploratória dos dados realizada, faz-se necessário o pré-processamento dos dados que, como dito anteriormente, foram a remoção das pontuações, stopwords e a normalização das strings, deixando todas em lowercase. Para realizar esse processamento foram utilizadas as seguintes funções e métodos:

```
# Remoção de pontuação
def dot_remove(x):
    try:
        x = ''.join(ch for ch in x if ch not in exclude)
    except:
        pass
    return x
```

```
# Tokenização
def tokenize(text):
    # Tokenização
    tokens = nltk.word_tokenize(text)

    # Stemização
    stems  = []
    for item in tokens:

stems.append(nltk.stem.snowball.SnowballStemmer('english').stem(item))
    return stems
```

```
# Removendo a pontuação
dt_train['text'] = dt_train['text'].apply(dot_remove).str.lower()
dt_test['text'] = dt_test['text'].apply(dot_remove).str.lower()

# Remoção das StopWords no processo de vectorização
stop_words = nltk.corpus.stopwords.words('english')

text_clf = Pipeline([# vectorize
                     ('vect', TfidfVectorizer(tokenizer=tokenize,
                                              stop_words = stop_words,
                                              ngram_range=(1, 2)))...])
```

Com o pré-processamento aplicado, nosso dataset agora fica como mostrado a seguir:

```
id  misogynous                                                text
0    0           0  me working remotely doing the chores doing gro...
1    1           1  imgilipcom divorce childrens wellbeing woman w...
2    2           1         a girl who shows a lot has little to offer
3    3           1  feminist we can do everything that men do men ...
4    4           1  rconservativememes posted by uundueinfluence 1...
5    5           1            this is not feminism 3 this is feminism
6    6           0     you have the personality of a carpeted kitchen
7    7           1  the entire halftime show was a form of female ...
8    8           1      hot blonde going thru a street no way in g...
```

3 Configuração experimental

Para a realização tanto da análise dos dados, quanto da realização do pré-processamento dos dados, treinamento, teste, verificação de métricas e desempenho dos algoritmos testados, foi utilizada a linguagem de programação Python que é executada por dentro da ferramenta Colaboratory do Google na versão 3.8. As bibliotecas utilizadas, são de maioria providas pelo sklearn, contudo, também foi utilizada a biblioteca pandas, csv, matplotlib, itertools, nltk e string, para realização das análise, tratamentos e coleta de resultados dos nossos datasets. A seguir, todas as bibliotecas que foram utilizadas nas análises:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
```

```
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn import svm
from itertools import zip_longest
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import string
```

4 Algoritmos utilizados

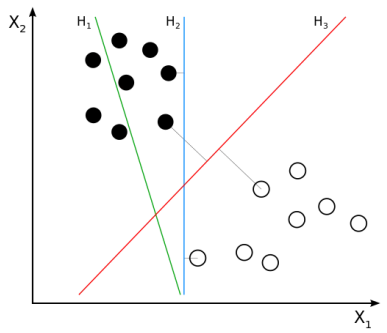
Os algoritmos utilizados no treinamento foram o Randon Forest, MLPClassifier e o SVM e comparados para verificar qual seria mais eficiente para a resolução do nosso problema.

No classificador RandomForest serão criadas várias árvores de decisão, onde em cada uma delas acontecem os seguintes passos:

1 - seleção aleatória de algumas features,
2 - seleção da feature mais adequada para a posição de nó raiz,
3 - geração dos nós filhos.

Já para o MLPClassifier ou Perceptron Multicamadas (PMC ou MLP — Multi Layer Perceptron) que é uma rede neural com uma ou mais camadas ocultas contendo um numero não determinado de neurônios, Tal camada é chamda de oculta pois não é possível prever a saída desejada nas camadas intermediárias. E para realiza o treinamento da rede MLP, o algoritmo comumente utilizado é o que realiza a retropropagação (Backpropagation).

O algoritimo SVM busca uma linha de separação entre duas classes distintas analisando os dois pontos, um de cada grupo, mais próximos da outra classe, ou seja, o SVM escolhe a reta, também chamada de hiperplano em maiores dimensões, entre dois grupos que se distancia mais de cada um (no caso abaixo, a reta vermelha).



Após encontrar essa reta, o programa será capaz de predizer a qual classe pertence um novo input ao verificar de qual lado da reta ele está.

Esses algoritmos de classificação podem ser instanciados da seguinte forma:

```
# Algoritimo RANDOM_FOREST.
text_clf = Pipeline([# vectorize
                     ('vect', TfidfVectorizer(tokenizer=tokenize,
                                              stop_words = stop_words,
                                              ngram_range=(1, 2))),
                     #classifier
                     ('clf', RandomForestClassifier(n_estimators=25,
max_depth=None, min_samples_split=2, random_state=3)),
                     ])



# Algoritimo MLPClassifier.
text_clf = Pipeline([# vectorize
                     ('vect', TfidfVectorizer(tokenizer=tokenize,
                                              stop_words = stop_words,
                                              ngram_range=(1, 2))),
                     #classifier
                     ('clf', MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(70, ), random_state=1, verbose=True)),
                     ])


# Algoritimo SVM.
text_clf = Pipeline([# vectorize
                     ('vect', TfidfVectorizer(tokenizer=tokenize,
                                              stop_words = stop_words,
                                              ngram_range=(1, 2))),
                     #classifier
                     ('clf', svm.SVC(kernel='linear')),
                     ])


# Treinamento
text_clf = text_clf.fit(x_train.text, y_train)
```

5 Resultados

Para análise das métricas e desempenho de cada classificador para o nosso conjunto de dados, foi utilizado o módulo metrics da biblioteca sklearn, com ele conseguimos demonstrar para nossos algoritmos a Precisão, Recall, F1-Score, Acurácia e Matriz de confusão, tudo isso através das seguintes funções:

```
# Separação dos dados para TESTE de algoritmos.
x_train, x_test, y_train, y_test = train_test_split(dt_train[['text']],
dt_train.misogynous, train_size=0.8, random_state=42)
```

```
# Verificação de Métricas para medir o desempenho da Classificação.
y_true, y_pred = y_test, text_clf.predict(x_test.text)
c = confusion_matrix(y_test, y_pred)
print(classification_report(y_true, y_pred))
print('\nAccuracy: {:.3f}%'.format(float(accuracy_score(y_test, y_pred)
* 100)))
print('\nConfusion Matrix:')
disp = ConfusionMatrixDisplay(confusion_matrix=c,
                              display_labels=text_clf.classes_)
disp.plot()
```

- A precisão é a capacidade do classificador não rotular uma amostra negativa como positiva.
- O recall é intuitivamente a habilidade do classificador em encontrar todas as amostras positivas.
- A pontuação F-1 pode ser interpretada como uma média harmônica ponderada da precisão e da revocação, onde uma pontuação F-beta atinge seu melhor valor em 1 e pior pontuação em 0. Os pesos da pontuação F-beta lembram mais do que a precisão por um fator de beta. beta == 1.0 significa que o recall e a precisão são igualmente importantes.
- Para a interpretação da Matriz de Confusão, levamos em consideração a seguinte análise:

  Elemento na posição (0,0)TP: representa a quantidade de classificações Verdadeiros Positivos, ou seja, a quantidade de vezes que o modelo acertou a predição positiva conforme os dados reais. No exemplo, o classificador previu corretamente X casos em que o resultado foi MISOGYNOUS(1).

  Elemento na posição (0,1)FN: representa a quantidade de classificações Falsos Negativos, ou seja, a quantidade de vezes que o modelo previu incorretamente um resultado como negativo. No exemplo, o classificador previu incorretamente Y casos

em que o resultado foi MISOGYNOUS(0), sendo que a predição correta deveria ser MISOGYNOUS(1).

Elemento na posição (1,0)FP: representa a quantidade de classificações Falsos Positivos, ou seja, a quantidade de vezes que o modelo previu incorretamente um resultado como positivo. No exemplo, o classificador previu incorretamente X casos em que o resultado foi MISOGYNOUS(1), sendo que a predição correta deveria ser MISOGYNOUS(0).

Elemento na posição (1,1)TN: representa a quantidade de classificações Verdadeiros Negativos, ou seja, a quantidade de vezes que o modelo acertou a predição negativa conforme os dados reais. No exemplo, o classificador previu corretamente X casos em que o resultado foi MISOGYNOUS(0).

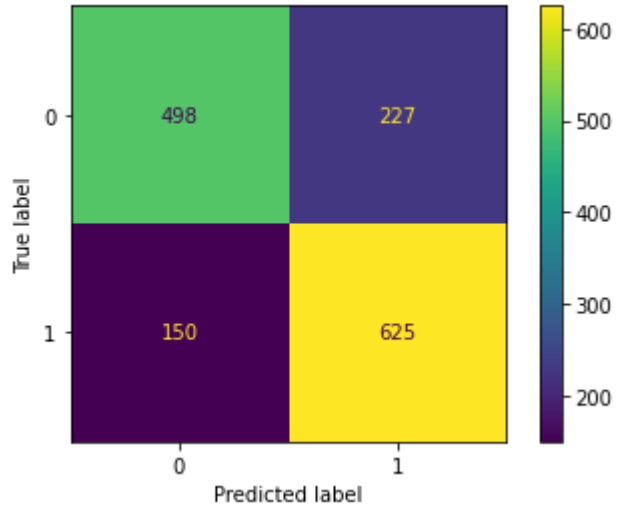
Logo abaixo, separado por algoritmos, temos os resultados obtidos após a aplicação de cada um dos classificadores apresentados anteriormente.

```
# Algoritmo RANDOM_FOREST.
```

	precision	recall	f1-score	support
0	0.77	0.69	0.73	725
1	0.73	0.81	0.77	775
accuracy			0.75	1500
macro avg	0.75	0.75	0.75	1500
weighted avg	0.75	0.75	0.75	1500

```
Accuracy: 74.867%
```

Confusion Matrix:

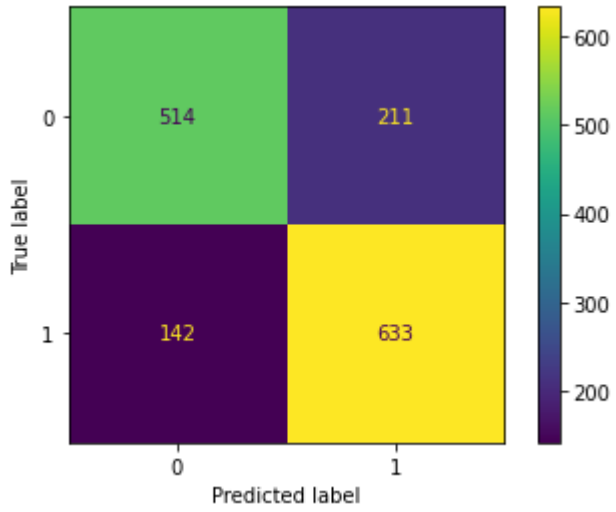


```
# Algoritmo MLPClassifier.
```

	precision	recall	f1-score	support
0	0.78	0.71	0.74	725
1	0.75	0.82	0.78	775
accuracy			0.76	1500
macro avg	0.77	0.76	0.76	1500
weighted avg	0.77	0.76	0.76	1500

```
Accuracy: 76.467%
```

Confusion Matrix:

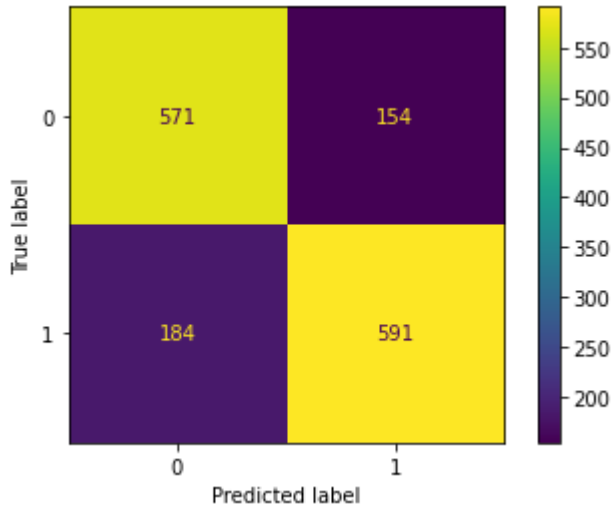



```
# Algoritmo SVM.
```

	precision	recall	f1-score	support
0	0.76	0.79	0.77	725
1	0.79	0.76	0.78	775
accuracy			0.77	1500
macro avg	0.77	0.78	0.77	1500
weighted avg	0.78	0.77	0.77	1500

```
Accuracy: 77.467%
```

Confusion Matrix:



Dado os resultados a escolha como algoritmo mais eficiente para a classificação dos dados é fácil, pois o SVM apresentou métricas bem mais eficientes do que o RandomForest e o MLPClassifier, ao submeter os algoritimos no Kaggle, na página da competição, os scores obtidos foram de 0.74865 para o RandonForest, 0.75699 para o MLP e por fim 0.78780 para o SVM, com essa confirmação, a escolha mais segura para nosso dataset dentre os modelos de classificação é o algoritmo SVM.

6 Referências bibliográficas

[1]Vídeos teóricos, slides e materiais disponibilizados pela professora.
[2]https://medium.com/luisfredgs/classificando-textos-com-machine-learning-e054ca7bf4e02
[3]https://medium.com/turing-talks/turing-talks-12-classifica%C3%A7%C3%A3o-por-svm-f4598094a3f1
[4]https://edisciplinas.usp.br/pluginfile.php/4308396/mod_resource/content/2/cb_14_rn_mlp_2.pdf
[5]https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9
[6]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier
[7] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[8] https://www.ibm.com/cloud/learn/random-forest
[9]https://dadosaocubo.com/analise-exploratoria-de-dados-com-python-parte-i/
[10]https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509